MDE4BBIS: A Framework to Incorporate Model-Driven Engineering in the Development of Blockchain-Based Information Systems

Amaral de Sousa, Victor; Burnay, Corentin

Published in:
2021 3rd International Conference on Blockchain Computing and Applications, BCCA 2021

DOI:
10.1109/bcca53669.2021.9657015

Publication date:
2021

Document Version
Early version, also known as pre-print

MDE4BBIS: A Framework to Incorporate Model-Driven Engineering in the Development of Blockchain-Based Information Systems

Victor Amaral de Sousa
PReCISE Research Center
Namur Digital Institute
University of Namur
Namur, Belgium
Email: victor.amaral@unamur.be

Corentin Burnay
PReCISE Research Center
Namur Digital Institute
University of Namur
Namur, Belgium
Email: corentin.burnay@unamur.be

***Abstract*—Model-driven engineering is a promising software development methodology that has been investigated in the context of blockchain-based information systems development. Many approaches propose to specify and generate individual components part of such systems' architectures using this methodology. In this paper, we provide a high-level overview of the different types of components that can be generated using model-driven engineering, and of the potential benefits that it could bring. We organize these findings in a framework called MDE4BBIS, which can help identifying opportunities to leverage model-driven engineering for different architectural layers in blockchain-based information systems, and promotes an integrated approach.**

***Index Terms*—blockchain, model-driven engineering, model-driven development, framework**

I. Introduction

In 2008, Bitcoin was introduced as a decentralized digital currency by Satoshi Nakamoto. The underlying technology used to record and validate transactions in an immutable, decentralized and distributed manner is referred to as blockchain technology. More recent platforms such as Ethereum added an additional layer to support smart contracts. These are software that can be deployed and executed on blockchain networks to implement business logic and manage transactions to the underlying database. Blockchain technology and smart contracts offer new opportunities and disruptive potential in a wide range of domains and industries. They are however complex and emerging technologies, which makes their development and the development of systems interacting with them particularly challenging. In the remainder of this paper, we refer to such systems as Blockchain-Based Information Systems (BBISs), and focus on platforms supporting smart contracts.

Model-Driven Engineering (MDE) is a software development methodology that allows generating models or executable software code from models used as input. The use of MDE for the development of BBISs has been investigated from different perspectives in various studies. For instance, [1] presents an approach to generate smart contracts handling asset registries and business processes from Business Process Model and Notation (BPMN) and asset registry models.

Existing work has investigated the use of MDE for the development of specific components of BBISs (e.g. smart contracts). However, to the best of our knowledge, there are no studies providing an overview of the different components that can be generated using MDE in the development of BBISs and of the overall benefits that it can bring. Such an overview could help organizations and researchers identifying opportunities to leverage MDE to improve their BBISs development.

To bridge this gap, we have reviewed existing work on the use of MDE for the development of BBISs. In particular, we have identified the types of artefacts generated as output of the approaches and the overall benefits that were brought by MDE in this context (section II).

Based on this analysis, we propose a first version of the Model-Driven Engineering for Blockchain-Based Information Systems (MDE4BBIS) framework (section III). It depicts a typical BBIS architecture and shows which components can be generated using MDE for the different architectural layers. This allows identifying opportunities to use MDE in the development of BBISs. To illustrate the use of the framework, we applied it to the case of B-MERODE, an approach to generate smart contracts supporting business processes [2].

This research allowed the identification of a number of future research directions. These concerns are discussed in section IV, which is followed by the general conclusions of the paper in section V.

II. The Role of MDE in the development of BBISs

The idea of using MDE in the development of BBISs is not new and has been proposed and applied in different areas. In this section, we provide an overview and summary of existing approaches and doing so, identify the types of artefacts that are generated as output of the MDE process and the resulting benefits. The results are summarized in Table I.

A. Smart Contracts

A first domain that is worth mentioning is Business Process Management (BPM). Blockchain technology and smart

contracts offer a number of opportunities and challenges for BPM, including the possibility to enforce, verify and automate the execution of business processes [3]. Solutions have been developed following these ideas, such as [1], [2], [4], [5]. In these contributions, new Domain-Specific Languages (DSLs) or well-recognized modeling languages such as Unified Modeling Language (UML) and BPMN are used (as input) for the generation of smart contracts that will enforce the modeled processes, and/or allow their monitoring and auditing.

The solutions discussed so far typically focus, in varying degrees, on the data being used in the process, on the parties involved in the process and on the sequence of activities that need to be executed. However, they put less emphasis on the automation of complex activities (part of such processes) using smart contracts. This is where other contributions come into play. For instance, [6] proposes to generate (parts of) smart contracts to enforce institutional rules that are captured using a human-readable format. Other examples include [7] which proposes to generate smart contracts enforcing rules specified in regulatory documents based on a format that allows to formalize the rules, along with domain knowledge and ontologies. Ontologies are also used in [8], which presents an executable formalization of the Resource-Event-Agent ontology that could be used to support economic transactions on blockchain-based systems. Besides this, [9] proposes to generate smart contracts acting as digital twins for the use of and interactions among cyper-physical system elements. A last example closer to the BPM discipline is [10], which proposes to generate smart contracts implementing decisions based on models created using the Decision Modeling Notation (DMN).

Overall, existing approaches demonstrate that models can be used to represent business logic (in various forms) and to generate smart contracts code implementing it. This can facilitate the specification and understanding of smart contracts and make them accessible to a broader audience [2], [4]–[6], [11]. This is a valuable element considering that domain experts and business analysts play a key role in the smart contracts development process. Also, considering that blockchain solutions typically involve multiple collaborating parties, if agreements are to be represented as smart contracts, it is key for all engaging entities to understand the contractual obligations that are implemented as smart contracts [6].

Once the business requirements have been described and modeled, they must be translated into smart contracts code. This task must be carried on carefully, and it is key to ensure that the implemented smart contracts are correct before they are deployed [2], [5], [9]. In this regard, using MDE to generate smart contracts can bring benefits in multiple areas.

First, before generating code based on models, the models themselves can be verified, and checking models is easier than checking code [5]. In some cases, verification can be done formally and automatically (e.g [2]). Once the models are checked, they can be used as basis for the implementation of the smart contracts. It is critical to have high-quality and correct implementations, especially considering the immutable character of smart contracts. A second benefit of MDE is that it can implement best practices for the smart contracts code and thereby reduce programming errors, other defects and their related costs [2], [5]–[7]. As a third benefit, it can also be used to optimize the resources that are required for smart contracts to run on blockchain infrastructures [12], which is a key consideration on platforms on which fees have to be paid depending upon resource utilization (e.g. Ethereum). On top of that, MDE in this context can also help avoiding common errors and ensuring protection against known attacks and vulnerabilities [11]. It can also be used to facilitate simulation of smart contracts, to ensure not only that their implementation is correct, but that the business logic is correctly specified, and that the business requirements are met [4], [5]. Finally, MDE facilitates reusability, which helps limiting redundant development effort [7]. This is particularly relevant considering the lack of people with adequate knowledge and skills, and the steep learning curve to follow for smart contracts and blockchain technology, as mentioned in [2].

B. Deployment Artefacts and Off-Chain Components

In other contributions, MDE is not used for the generation of smart contracts, but for other components of a BBIS. Among others, [13] proposed an approach to generate the configuration files required for the deployment of blockchain platforms based on models capturing the essence of a deployment design (e.g. blockchain platform, consensus algorithm, number of nodes). Other approaches propose the generation of off-chain components allowing interaction with smart contracts and/or the generation of testing code or testing applications for (generated) smart contracts (e.g. [4]).

Smart contracts play an essential role in most BBISs. However, they require a running infrastructure for their deployment and execution, and connectors together with end-user applications for their use. By generating these components using MDE, we can further accelerate and facilitate the testing and deployment of BBISs as less manual and technical work is required for each iteration.

In particular, there is a lack of modeling means for the deployment of BBISs, that [13] addresses. This is particularly relevant in the blockchain context as there are multiple deployment environments, and as multiple blockchain platforms are sometimes compared for the deployment of a solution. Using MDE to generate deployment files from deployment models could help targeting multiple platforms while ensuring consistency between the deployment models and the actual deployment [13].

TABLE I
ROLE OF MDE IN THE DEVELOPMENT OF BBISS: SUMMARY

Input Model Type	Generated Artefacts	Studies
Technical design model	Deployment artefacts	[13], [14]
Business logic model	Smart contracts	[2], [4]–[7], [9]
Business logic model	Blockchain connectors	[4], [11], [13]
Business logic model	End-user applications	[4]

Complementary to this approach, creating blockchain network technical design models as input for the generation of network simulators can facilitate the testing and evaluation of various (possibly new) designs [14]. It can also facilitate sharing and reusability of design models, along with their evaluation results in the research community [14]. Finally, it will allow cost- and time-effective study of blockchain networks in various operational scenarios [14].

Using models as input to generate (parts of) BBISs using MDE allows early and easier (possibly automated) testing and prototyping, which, in turn, improves software quality as well as cost- and time-efficiency. By making BBISs development and testing faster, cheaper, easier and more accessible, MDE can help foster adoption of blockchain technology [2], [5], [6].

III. The MDE4BBIS Framework

In the previous section, the role of MDE in the development of BBISs has been discussed, and 3 groups of artefacts were identified: artefacts for (technical) blockchain deployment, smart contracts and off-chain components.

On this basis, we propose a first version of the Model-Driven Engineering for Blockchain-Based Information Systems (MDE4BBIS) framework, which shows how MDE approaches could be integrated for the specification and implementation of end-to-end BBISs.

Such a framework could help organizations identifying opportunities to further integrate MDE practices in their BBISs development initiatives to leverage its benefits. While the framework can suggest opportunities, these would still need to be evaluated separately, considering the specificities of the application context and its particular challenges.

A. Framework Description

The framework that we propose is depicted on Fig. 1. Its structure is derived from [15]. The authors present 3 different model layers, with automated transformations between them. The first layer that we consider consists in platform-independent models representing analysis and design of the system. The second layer is made of platform-specific models that are described as more detailed design models. Finally, the last layer is made of implementation and runtime models (executable code being such a model). Based on this, we organize the framework around a 3-steps process: (1) the modeling that consists in creating the platform-independent models, (2) transformation of these models into platform-specific models and finally, (3) the execution of functional components (referring to the implementation-specific models).

The first step consists in capturing various requirements (possibly relating to various components of BBISs) in a (set of) model(s) that are platform-independent. In section II, we discussed different types of models. Business logic models (A) help capturing business-oriented requirements, in two main forms: process models (A*) or activity models (A**). The former allow capturing details about different components of business processes (e.g. data structures and/or activity sequences) while the latter focus on the representation of specific activities that can be part of such processes (e.g. a specific decision that is part of a process). We assume that the data models are (either implicitly of explicitly) described as part of the process and activity models. Complementary to business logic models, technical design models (B) allow capturing requirements related to the blockchain (technical) infrastructure (e.g. number of nodes and consensus algorithm). We chose to classify such models as platform-independent, since they are not always targeting one specific blockchain platform, but rather allow building, configuring and deploying a variety of such platforms. In case where the goal is to capture the platform requirements by focusing on one specific case (e.g. Corda or Hyperledger Fabric), then the models capturing these details should be considered as platform-specific.

Once a first set of models is available, it can be transformed if necessary into other models, that can be platform-independent or platform-specific. Finally, the intermediary (or base models) can be transformed into functional components of the BBIS being designed. The intermediary models that are used along with the model-to-model or model-to-code transformation rules are out of the scope of the present paper.

In Fig. 1, we depict typical components present in the architecture of a BBIS. First we have a set of off-chain components that include end-user applications (1) and the blockchain connector (2), allowing interaction between end-user applications and smart contracts (3). The smart contracts are deployed on a running blockchain infrastructure (4), and both are grouped as on-chain components.

In the framework, we identified at a conceptual level the types of models can be used to generate different components in the BBIS architecture. From business logic models, existing approaches generate (parts of) end-user applications (A1), blockchain connectors (A2) and smart contracts (A3). Finally, from technical design models, it is possible to generate running blockchain infrastructures (B4).

While these possibilities offer a number of potential benefits, the integration of multiple approaches is seldom discussed in existing work, but could further increase the benefits, considering the relationships and dependencies between the elements considered in the traditional architecture. Starting from business logic models, different components can be generated. However, without integration between the approaches, it is likely that the models used for the different artefacts will be expressed in different, non-integrated languages. Furthermore, it is also likely that additional (manual) effort will be required to connect the different components together. In this context, using a language that allows, using different views, to model and generate implementations for multiple components that are integrated together, could further increase the benefits of the MDE approach.

Similarly, as suggested in [13], multiple benefits could be drawn from the integration of MDE approaches for the deployment of blockchain infrastructures and the generation of smart contracts. Indeed, the two can sometimes be linked together. For instance, when modeling cross-organizational business processes to be supported by blockchain technology,

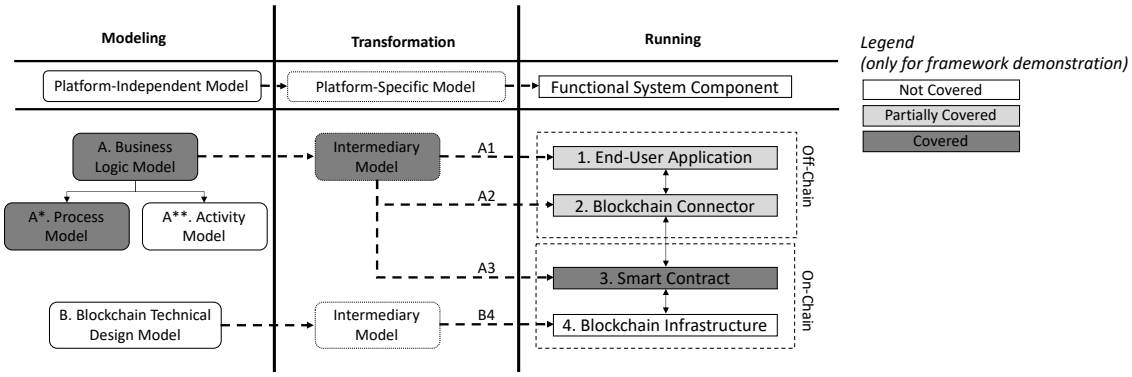

Fig. 1. MDE4BBIS Framework Overview

it is common to identify the different types of participants in the business collaboration (e.g. [2]). This could be used as input to define certain network topologies for the underlying blockchain architecture (e.g. x validation nodes by participant). Overall, the more components can be described using appropriate and integrated models, the easier it will be to create and maintain these models and to generate blockchain-based solutions as efficiently and easily as possible.

B. Framework Demonstration

In [2], we propose a conceptual (and graphical) modeling language, B-MERODE, that can be used to specify blockchain-supported cross-organizational business processes. The language follows a consistency by construction approach for the models, for which a number of properties can be formally and automatically verified. In the proposed framework, B-MERODE allows to create platform-independent business logic models that are then verified and transformed into smart contracts (A3). At the moment, the smart contracts are generated to run on the Hyperledger Fabric platform and all the generated smart contracts offer a common interface.

In the remainder of this section, we apply the MDE4BBIS framework to the case of B-MERODE to demonstrate how it can be used (see Fig. 1 for a high-level representation).

While B-MERODE allows modeling overall processes and supporting their execution on the blockchain, complex decisions that could be automated through smart contracts cannot always be represented using the available constructs. Therefore, it would be interesting to evaluate to which extent activity models (A**) could be incorporated in the approach to allow the execution of business processes and the automation of particular activities (e.g. using DMN models).

Once smart contracts have been generated, they still need to be deployed onto running blockchain infrastructures, which must be configured and set up manually. By using technical design models, it would be possible to generate the configuration files that are required, and to automatically deploy running infrastructures (B4). Furthermore, in a context where blockchain is used to support business processes, some elements that are needed in the infrastructure could be defined based on the existing models that are used in B-MERODE. For instance, when describing a process using this language, the different types of participants need to be identified, as they have different permissions. Based on this, it would be possible to generate deployment and configuration files with, for instance, a certain number of nodes per participant (or participant type), and an initial number of participants of each type. This would further facilitate the deployment and testing of solutions generated using B-MERODE, which currently requires a manual setup of the infrastructure.

Assuming that the technical infrastructure and smart contracts are available, we still need to test the system and to develop end-user applications and connectors for them to use the smart contracts. In the current state, B-MERODE does not generate end-user applications (A1) or blockchain connectors (A2). With the information available in the models, and considering how generated smart contracts are structured, it would be possible to either use template-based blockchain connectors, or to generate connectors that would allow invoking smart contracts. Regarding end-user applications, it is worth mentioning that B-MERODE is based on MERODE [16], an enterprise information systems engineering method that is not originally focused on blockchain technology. Using the MERODE language and tools, it is possible to generate working application prototypes [16], testing suites [17] and user interfaces [18]. However, not all of these possibilities have yet been incorporated for the generation of blockchain-based information systems. By performing such an integration, it would be possible, starting from a set of models, to generate smart contracts and end-user applications allowing interaction with the generated smart contracts.

By integrating the different perspectives (A1, A2, A3, B4), it would be possible to generate end-to-end working prototypes of BBISs, which would allow fully leveraging the benefits of a MDE approach. It would also be possible to deploy a same set of smart contracts on various blockchain network technical designs to have design-specific evaluations.

IV. Discussion

In this paper, we reviewed the literature to identify the role that MDE could play in the development of BBISs, and the benefits that it could bring. To do so, we did not follow

a systematic procedure. Therefore, we cannot claim that the framework is exhaustive. It provides however a first overview and could be used as a starting point for further and more systematic research.

The discussion held in this paper is kept at a conceptual and high-level to allow the identification of general research directions, that may either be domain-specific or more generic. For instance, questions such as which components of business processes should be captured and how they can be translated into smart contracts are relevant from a BPM perspective, and would have specific answers. Questions such as how to generate blockchain network deployment files from models are however more generic considerations. In both cases, answering these questions in various domains would constitute a lead for further research.

In this paper, we have adopted a single-chain view of BBISs. However, as mentioned in [19], more and more solutions are relying on multiple blockchain platforms as part of their infrastructure. MDE can also bring benefits in this area.

An additional consideration is the fact that this paper offers rather positive view on MDE as a solution to solve multiple challenges in the development of BBISs. However, incorporating MDE in the development process brings on a number of issues and challenges on its own, which are not discussed in this paper. Studies on the challenges in the incorporation of MDE in specific domains or for particular components also constitute a general lead for further research.

Finally, in the presentation of the framework, we discussed the interactions and integration between models at different architectural levels. It would be interesting to investigate such interactions and integration across and within various domains. For instance, [6] proposes to generate elements of smart contracts to enforce institutional agreements. The verification and enforcement of such agreements, on a more global level, is embedded in business processes involving multiple parties. Investigating how this approach could be combined with a solution generating smart contracts to execute business processes and automate particular process activities could yield interesting results.

V. Conclusions

In this paper, we have provided a first overview of the role that MDE could play in the development of BBISs, and of the potential benefits that it could bring.

Based on this, we proposed a first attempt to structure and identify opportunities to use MDE in the development process in the form of a framework called MDE4BBIS. This framework identifies the elements of a typical BBIS architecture that can be generated from different sets of models. It also discusses the interest of integrating different models together in other to further leverage the benefits that MDE can bring at different individual levels. A high-level example of how the framework can be used was presented as demonstration.

Overall, MDE can facilitate the understanding, specification, implementation, testing, validation and deployment of BBISs, while being more cost- and time-efficient than fully manual development, and leading to higher-quality solutions. Such benefits would ultimately foster adoption of blockchain technology and lead to more robust systems. Integrating MDE at multiple architectural levels in the development process would help further leveraging the benefits of MDE.

As further research leads, we propose to use this study as a starting point for a systematic research in order to have a more comprehensive and representative framework. We also propose to assess the benefits, and in particular the challenges (that were not discussed in the present paper) of using MDE at different architectural levels, in specific domains. It would also be interesting to investigate how research in different domains could be integrated in a broader MDE perspective. Finally, this study was conducted with a single-chain view, while modern multi-chain solutions might also benefit from MDE.

References

[1] Q. Lu, A. B. Tran, I. Weber, H. O'Connor, P. Rimba, X. Xu, M. Staples, L. Zhu, and R. Jeffery, "Integrated model-driven engineering of blockchain applications for business processes and asset management," *Software: Practice and Experience*, vol. 51, no. 5, pp. 1059–1079, 2021.

[2] V. Amaral de Sousa, C. Burnay, and M. Snoeck, "B-merode: A model-driven engineering and artifact-centric approach to generate blockchain-based information systems," in *Advanced Information Systems Engineering*, S. Dustdar, E. Yu, C. Salinesi, D. Rieu, and V. Pant, Eds. Cham: Springer International Publishing, 2020, pp. 117–133.

[3] J. Mendling, I. Weber, W. V. D. Aalst, J. V. Brocke, C. Cabanillas, F. Daniel, S. Debois, C. D. Ciccio, M. Dumas, S. Dustdar, A. Gal, L. García-Bañuelos, G. Governatori, R. Hull, M. L. Rosa, H. Leopold, F. Leymann, J. Recker, M. Reichert, H. A. Reijers, S. Rinderle-Ma, A. Solti, M. Rosemann, S. Schulte, M. P. Singh, T. Slaats, M. Staples, B. Weber, M. Weidlich, M. Weske, X. Xu, and L. Zhu, "Blockchains for Business Process Management - Challenges and Opportunities," *ACM Transactions on Management Information Systems (TMIS)*, vol. 9, no. 1, pp. 1–16, 2018.

[4] M. Skotnica, J. Klicpera, and R. Pergl, "Towards model-driven smart contract systems - code generation and improving expressivity of smart contract modeling," Czech Technical University, Tech. Rep., 2020.

[5] N. Sánchez-Gómez, L. Morales-Trujillo, and J. Torres-Valderrama, "Towards an approach for applying early testing to smart contracts," in *WEBIST 2019 - Proceedings of the 15th International Conference on Web Information Systems and Technologies*, 2019, pp. 445–453.

[6] C. K. Frantz and M. Nowostawski, "From institutions to code: Towards automated generation of smart contracts," in *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2016, pp. 210–215.

[7] O. Choudhury, M. Dhuliawala;, N. Fay, N. Rudolph, I. Sylla, N. Fairoza, D. Gruen, and A. Das, "Auto-translation of Regulatory Documents into Smart Contracts," in *IEEE Blockchain Initiative*, 9 2018, pp. 1–5.

[8] W. P. R. Laurier, S. Horiuchi, and M. Snoeck, "An executable axiomatization of the REA2 ontology," *Journal of Information Systems*, 2021.

[9] P. Garamvölgyi, I. Kocsis, B. Gehl, and A. Klenik, "Towards Model-Driven Engineering of Smart Contracts for Cyber-Physical Systems," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2018, pp. 134–139.

[10] S. Haarmann, K. Batoulis, A. Nikaj, and M. Weske, "DMN Decision Execution on the Ethereum Blockchain," in *Advanced Information Systems Engineering*, 2018, pp. 327–341.

[11] X. Xu, I. Weber, M. Staples, and I. Weber, *Architecture for Blockchain Applications*. Springer, Cam, 2019.

[12] L. García-Bañuelos, A. Ponomarev, M. Dumas, and I. Weber, "Optimized execution of business processes on blockchain," in *Business Process Management*, J. Carmona, G. Engels, and A. Kumar, Eds. Cham: Springer International Publishing, 2017, pp. 130–146.

[13] T. Górski and J. Bednarski, "Applying Model-Driven Engineering to Distributed Ledger Deployment," *IEEE Access*, vol. 8, pp. 118 245–118 261, 2020.

[14] S. Liaskos, T. Anand, and N. Alimohammadi, "Architecting blockchain network simulators: a model-driven perspective," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, vol. 00, 2020, pp. 1–3.

[15] A. Metzger, "Introduction," in *Model-Driven Software Development*, S. Beydeda, M. Book, and V. Gruhn, Eds. Berlin, Heidelberg: Springer, 2005, ch. 1, pp. 1–16.

[16] M. Snoeck, *Enterprise Information Systems Engineering, The MERODE Approach*. Cham: Springer International Publishing, 2014.

[17] B. Marín, S. Alarcón, G. Giachetti, and M. Snoeck, "Tescav: An approach for learning model-based testing and coverage in practice," in *Research Challenges in Information Science*, F. Dalpiaz, J. Zdravkovic, and P. Loucopoulos, Eds. Cham: Springer International Publishing, 2020, pp. 302–317.

[18] J. Ruiz, G. Sedrakyan, and M. Snoeck, "Generating user interface from conceptual, presentation and user models with jmermaid in a learning approach," in *Proceedings of the XVI International Conference on Human Computer Interaction*, ser. Interacción '15. New York, NY, USA: Association for Computing Machinery, 2015.

[19] G. Falazi, U. Breitenbücher, F. Daniel, A. Lamparelli, F. Leymann, and V. Yussupov, "Smart Contract Invocation Protocol (SCIP): A Protocol for the Uniform Integration of Heterogeneous Blockchain Smart Contracts," in *Advanced Information Systems Engineering*. Cham: Springer International Publishing, 2020, pp. 134–149.